AMENDMENTS TO THE CLAIMS:

1. (currently amended) A method of selecting an optimal command node in a computing device having ~~insert~~ a first queue containing a command node and a ~~sorted~~ second queue containing another command node, the method comprising steps of:

(a) determining if the command node in the ~~insert~~ first queue collides with the command node in the ~~sorted~~ second queue:

(a)(i) if the command node in the ~~insert~~ first queue does not collide with the command node in the ~~sorted~~ second queue, then moving the command node in the ~~insert~~ first queue from the ~~insert~~ first queue into the ~~sorted~~ second queue;

(a)(ii) if the command node in the ~~insert~~ first queue collides with the command node in the ~~sorted~~ second queue, then correcting the collision~~; and~~

~~(b) sorting the sorted queue according to a predetermined routine to determine an optimal command node and selecting the optimal command node~~.

2. (currently amended) ~~The method of claim 1~~ A method of selecting an optimal command node in a computing device having a first queue containing a command node and a second queue containing another command node, the method comprising steps of:

(a) determining if the command node in the first queue collides with the command node in the second queue:

(a)(i) if the command node in the first queue does not collide with the command node in the second queue, then moving the command node in the first queue from the first queue into the second queue;

(a)(ii) if the command node in the first queue collides with the command node in the second queue, then correcting the collision; and

(b) sorting the second queue according to a predetermined routine to determine an optimal command node and selecting the optimal command node, wherein the computing device is a disc drive device having a magnetic disc, wherein each command node includes information defining a range of addresses on the magnetic disc, and wherein the determining step (a) comprises, determining if the range of addresses defined by the command node in the ~~insert~~ first queue overlaps the range of addresses defined by the command node in the ~~sorted~~ second queue, is a subset of the range of addresses defined by the command node in the ~~sorted~~ second queue, or is a superset of the range of addresses defined by the command node in the ~~sorted~~ second queue.

3. (currently amended) The method of claim ~~1~~ 33, wherein the predefined routine comprises a Rotational Positioning Sorting (RPS) algorithm to optimize rotational latency.

4. (currently amended) ~~The method of claim 1~~ A method of selecting an optimal command node in a computing device having a first queue containing a command node and a second queue containing another command node, the method comprising steps of:

(a) determining if the command node in the first queue collides with the command node in the second queue:

(a)(i) if the command node in the first queue does not collide with the command node in the second queue, then moving the command node in the first queue from the first queue into the second queue;

(a)(ii) if the command node in the first queue collides with the command node in the second queue, then correcting the collision; and

(b) sorting the second queue according to a predetermined routine to determine an optimal command node and selecting the optimal command node , wherein the collision correction step (a)(ii) comprises the steps of:

(a)(ii)(A) determining if the command node in the ~~insert~~ first queue fully overlaps the command node in the ~~sorted~~ second queue;

(a)(ii)(A)(1) if the command node in the ~~insert~~ first queue fully overlaps the command node in the ~~sorted~~ second queue, then extracting the overlapped command node from the ~~sorted~~ second queue and moving the command node from the ~~insert~~ first queue into ~~sorted~~ the second queue;

(a)(ii)(A)(2) if the command node in the ~~insert~~ first queue does not fully overlap the command node in the ~~sorted~~ second queue ~~not~~, then determining if the command node in the ~~insert~~ first queue is a subset of the command node in the ~~sorted~~ second queue:

(a)(ii)(A)(2)(a) if the command node in the ~~insert~~ first queue is not a subset of the command node in the ~~sorted~~ second queue, then truncating the overlapped command node in the ~~sorted~~ second queue such that it no longer overlaps the command node in the ~~insert~~ first queue and moving the command node in the ~~insert~~ first queue from the ~~insert~~ first queue into the ~~sorted~~ second queue.

5. (currently amended) The method of claim 1, wherein the act of moving the command node in the ~~insert~~ first queue from the ~~insert~~ first queue into the ~~sorted~~ second queue comprises extracting the

command node from the ~~insert~~ first queue and inserting the extracted command node into the ~~sorted~~ second queue.

6. (Original) A computer-readable media having computer-executable instructions for performing the steps recited in claim 1.

7. (Original) A method of populating a ready queue in disc drive computing device with read/write command nodes, the disc drive computing device having an insert queue containing a command node and a sorted queue containing another command node, the method comprising steps of:

(a) determining if the ready queue is empty:

(a)(i) if the read queue is empty, moving the command node from the insert queue into the ready queue;

(a)(ii) if the ready queue is not empty, determining if the command node in the insert queue collides with the command node in the sorted queue:

(a)(ii)(A) if the command node in the insert queue collides with the command node in the sorted queue:

(a)(ii)(A)(1) determining if the command node in the insert queue fully overlaps the command node in the sorted queue:

(a)(ii)(A)(l)(a) if the command node in the insert queue fully overlaps the command node in the sorted queue:

(a)(ii)(A)(1)(a)(i) removing the command node in the sorted queue from the sorted queue; and

(a)(ii)(A)(1)(a)(ii) moving the command node in the insert queue from the insert queue into the sorted queue;

(a)(ii)(A)(1)(b) if the command node in the insert queue does not fully overlap the command node in the sorted queue, determining if the command node in the insert queue is a subset of the command node in the sorted queue;

(a)(ii)(A)(1)(b)(i) if the command node in the insert queue is not a subset of the command node in the sorted queue:

(a)(ii)(A)(1)(b)(i)(A) truncating the command node in the sorted queue such that a collision no longer exists, and

(a)(ii)(A)(1)(b)(i)(B) moving the command node in the insert queue from the insert queue into the sorted queue;

(a)(ii)(B) if the command node in the insert queue does not collide with the command node in the sorted queue, moving the command node in the insert queue from the insert queue into the sorted queue;

(a)(ii)(C) sorting the sorted queue according to a predetermined method to determine the optimal command node; and

(a)(ii)(D) moving the optimal command node into the ready queue.

8. (Original) A computer-readable media having computer-executable instructions for performing the steps recited in claim 7.

9. (Original) The method of claim 7, wherein the moving step (a)(ii)(A)(1)(a)(ii) comprises, extracting the command node in the insert queue from the insert queue and inserting the extracted command node into the sorted queue.

10. (Original) The method of claim 7, wherein the insert queue, the sorted queue, and the ready queue each comprise a doubly-linked list, and wherein the command node in the insert queue is the command node at the head of the insert queue, and the moving steps (a)(ii)(A)(1)(a)(ii) and (a)(ii)(A)(1)(b)(i)(B) comprise, extracting the command node in the insert queue from the head of the insert queue and inserting the extracted command node into the tail of the sorted queue.

11. (Original) The method of claim 7, wherein the predetermined method of sorting the sorted queue comprises a Rotational Positioning Sorting (RPS) algorithm.

12. (Original) A method for managing a command node in a computing system having a microprocessor, an insert queue, a sorted queue, and a ready queue, comprising:

(a) inserting the command node into the insert queue;

(b) determining if the ready queue is empty:

(b)(i) if the ready queue is empty, extracting the command node from the insert queue and inserting the command node in the ready queue,

(b)(ii) if the ready queue is not empty, determining if the command node collides with any command in the sorted queue:

(b)(ii)(A) if the command node does not collide with any command in the sorted queue, extracting the command node from the insert queue and inserting the command node in the sorted queue;

(b)(ii)(B) if the command node collides with any command in the sorted queue:

(b)(ii)(B)(1) correcting the collision;

(b)(ii)(B)(2) extracting the command node from the insert queue; and

(b)(ii)(B)(3) inserting the command node in the sorted queue;

(b)(iii)(C) determining if the command node is the optimal command node in the sorted queue, if the command node is the optimal command node in the sorted queue extracting the command node from the sorted queue and inserting the command node in the ready queue.

13. (Original) The method of claim 12, wherein before inserting step (a) the command node is extracted from a free list and populated with command data.

14. (Original) The method of claim 12, wherein the computing system further includes an active queue, the method further comprising:

(c) determining if the active queue is full, if the active queue is not full extracting the command node from the ready queue and inserting the command node in the active queue.

15. (Original) The method of claim 14, further comprising:

(d) determining if a command node is being requested by the microprocessor, if a command node is being requested by the microprocessor extracting the command node from the active queue and inserting the command node into a free list.

16. (Original) The method of claim 12, wherein the insert queue, the sorted queue, and the ready queue each comprises a doubly-linked list of command nodes.

17. (Original) The method of claim 12, wherein the computing system comprises a disc drive microprocessor and an associated memory and where the command node comprises a command node for implementing a read/write command in the disc drive.

18. (Original) Computer-readable media having stored thereon:

an insert queue;

a sorted queue;

a ready queue;

a plurality of queue managers, each queue manager comprising microprocessor- executable code operable for directing a microprocessor, the plurality of queue managers including:

a command queue manager operable for populating command nodes with command data and for inserting populated command nodes into the insert queue;

an insert queue manager operable for determining whether a command node within the insert queue collides with a command node in the sorted queue, for correcting any collisions between a command node within the insert queue and a command node in the sorted queue, and for extracting a non-colliding command node from the insert queue and inserting the non-colliding command node into the sorted queue;

a sorted queue manager operable for selecting an optimal command node from the sorted queue in accordance with a predefined sorting scheme; and

a scheduler for scheduling the execution of the queue managers in a microprocessor.

19. (Original) The computer-readable media of claim 18, wherein the sorted queue manager is further operable for extracting the selected command node from the sorted queue and inserting the selected command node into the ready queue.

20. (Original) The computer-readable media of claim 18, further having stored thereon an active queue and a ready queue manager, the ready queue manager being operable for extracting a preferred command node from the ready queue and for inserting the extracted preferred command node into the active queue.

21. (Original) The computer-readable media of claim 20, further having stored thereon a free list queue and an active queue manager operable for extracting a requested command node from the active queue and for inserting the requested command node into the free list queue.

22. (Original) The computer-readable media of claim 17, wherein the insert queue, the sorted queue, and the ready queue each comprises a doubly-linked list of command nodes, and wherein each of the command nodes includes data for implementing a read/write command in a microprocessor of a disc drive device.

23. (currently amended) A system for managing command nodes in a computing device comprising:

a microprocessor;

computer-readable media;

a data structure stored on the computer-readable media, the data structure comprising:

~~an insert~~ a first queue comprising a command node;

a ~~sorted~~ second queue comprising a command node selected from the ~~insert~~ first queue, and

a ~~ready~~ third queue comprising a command node selected from the ~~sorted~~ second queue according to an predefined optimization scheme.

24. (currently amended) The system of claim 23, wherein the ~~sorted~~ second queue includes only non-colliding command nodes.

25. (currently amended) The system of claim 24, further including ~~an active~~ a fourth queue including a command node selected from the ~~ready~~ third queue.

26. (currently amended) The system of claim 25, wherein the ~~active~~ fourth queue comprises ~~includes~~ two command nodes both of which are accessible by the microprocessor.

27. (currently amended) The system of claim 25, further including a ~~free list~~ fifth queue from which command nodes have been extracted, populated with command data, and inserted into the ~~insert~~ first queue.

28. (Original) The system of claim 23, wherein each of the queues is structured as a doubly-linked list having a head and a tail.

29. (Original) The system of claim 28, wherein each queue has an associated head pointer which points the command node at the head of the associated queue and an associated tail pointer which points the command node at the tail of the associated queue.

30. (Original) The system of claim 29, wherein the microprocessor includes a plurality of registers and wherein each of the head pointers and each of the tail pointers is stored in an associated register in the microprocessor.

31. (currently amended) A queue processing system for managing a plurality of command node queues in a data storage device comprising:

a microprocessor; and

a queue processing means for controlling the position and flow of command nodes within and ~~through~~ directly between the plurality of command queues.

32. (newly added) The queue processing system of Claim 31, wherein at least one of the command nodes is repositioned from one of the plurality of command node queues to another of the plurality of command node queues upon a determination and correction of a collision between the at least one command node and a command node in the another of the plurality of command node queues.

33. (newly added) The method of Claim 1, further comprising the steps of:

sorting the second queue according to a predetermined routine to determine an optimal command node; and

selecting the optimal command node.